1. (Twice amended) In a computer system in which a first thread and a second thread of a user application execute concurrently in a common address space, a method of processing an application event in response to the detection of said application event by said first thread, comprising the steps of:

said first thread, in response to detecting said application event:

sending a quiesce event [from said first thread] to said second thread [in response to the detection of said application event by said first thread] to cause said second thread to quiesce; and

suspending execution [of said first thread] until said second thread has quiesced in response to the quiesce event sent to that thread; and

said second thread, in response to receiving said quiesce event:

determining whether it is holding any resource required by another thread;

quiescing only if it determines that it is not holding any resource required by another thread; and

upon quiescing, resuming execution of said first thread to process said application event [when said second thread has quiesced in response to the quiesce event sent to that thread].

11. (Twice amended) The method of Claim [10, further comprising the step of releasing] 1 wherein said second thread releases any resource required by another thread that is held by said second thread before quiescing said second thread.

12. (Twice amended) In a computer system in which a first thread and a second thread of a user application execute concurrently in a common address space, a method of processing an application event in response to the detection of said application event by said first thread, comprising the steps of:

said first thread, in response to detecting said application event:

sending a suspension event [from said first thread] to said second thread [in response to the detection of said application event by said first thread] to cause said second thread to suspend; and

suspending execution [of said first thread] until said second thread has suspended in response to the suspension event sent to that thread;

said second thread, in response to receiving said quiesce event:

determining whether it is holding any resource required by another thread;

quiescing only if it determines that it is not holding any resource required by another thread; and

upon quiescing, resuming execution of said first thread to process said application event [when said second thread has suspended in response to the suspension event sent to that thread]; and

said first thread resuming said second thread following the processing of said application event by said first thread.

Entry of this amendment and reconsideration of the application as amended are respectfully requested.

Remarks

The specification has been amended on page 5 to change "effected" to "affected", which is more appropriate to the context.

Claims 1 and 12 have been amended to recite which thread performs each step. Claims 1 and 12 have been further amended to incorporate the limitations of Claims 9 and 10, which have been cancelled. Claim 11, formerly dependent on Claim 10, has been amended to depend on Claim 1.

Claims 1 and 12 as amended each recite that a first thread, in response to detecting an application event, sends a quiesce event to a second thread to cause it to quiesce and suspends execution until the second thread has quiesced in response to the quiesce event sent to that thread. In